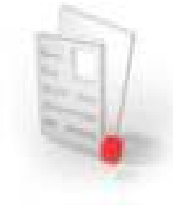
California Enterprise Architecture Program



SOA *White Paper*

SOA Service Patterns

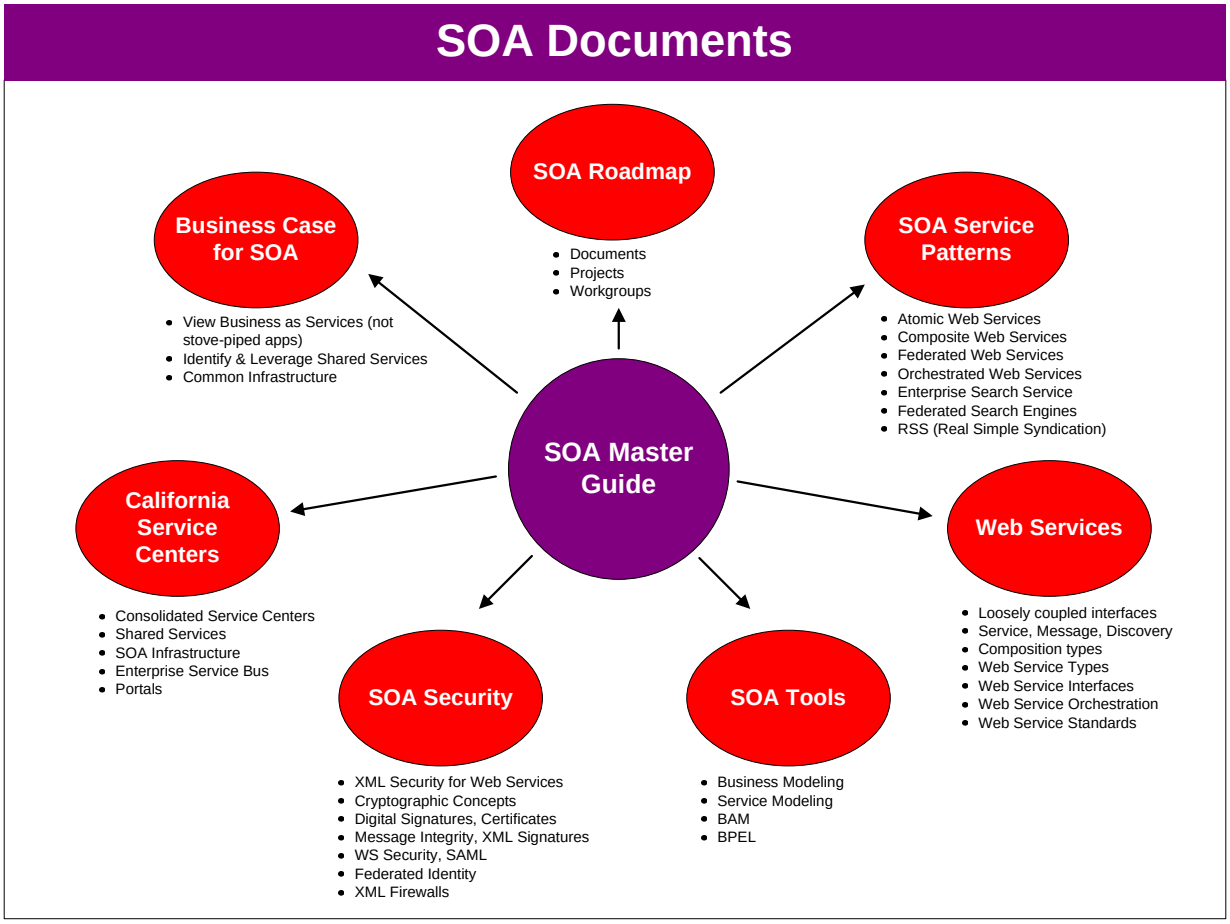
Revision History

06/22/2006 Original Draft

Table of Contents

SOA Documents

The service oriented architecture advocated by the California Enterprise Architecture Program is organized into a set of interrelated documents. A master guide serves as the "jumping off point" and describes in an overview fashion the key parts of SOA.

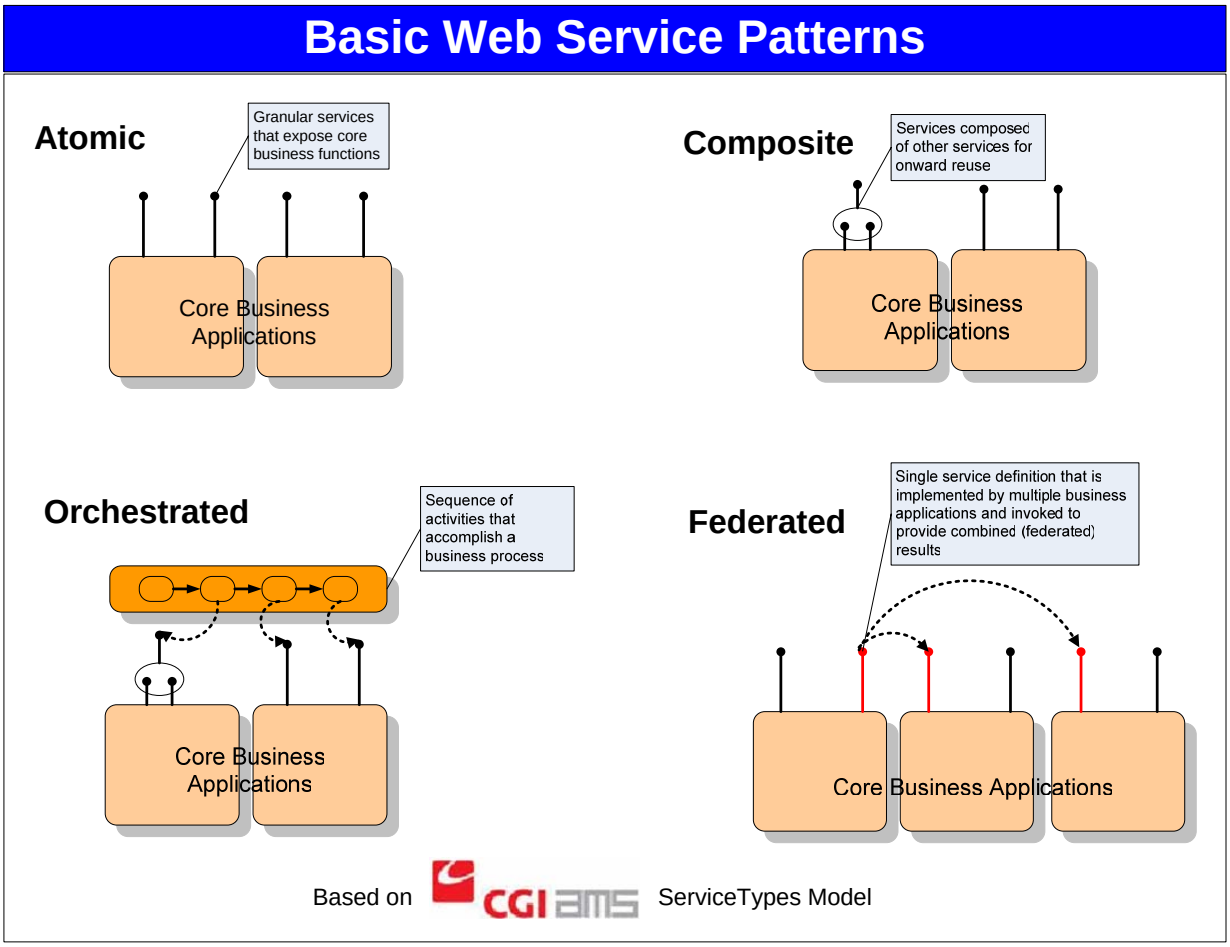


There are six white papers planned to address in depth details of SOA. This whitepaper is *SOA Service Patterns.*

QuickView – SOA Service Patterns

Web service patterns are architecture design patterns that illustrate different ways that you can use web services. Design patterns incorporate standards and best practice techniques that generally result in more consistent applications.

This white paper focuses on four basic web service patterns that can serve as the blueprint for any type of application using web services.



Additionally, this whitepaper describes three topic specific patterns. Two of them suggest ways to handle enterprise search and the third illustrates using the industry standard RSS (Real Simple Syndication) for publishing practically any type of public information.

Web services are simply business applications packaged with web service interfaces. Because these interfaces are based on XML messaging, they are highly interoperable meaning they hide the details of different programming languages and different hardware and software platforms. They simply communicate by exchanging XML messages.

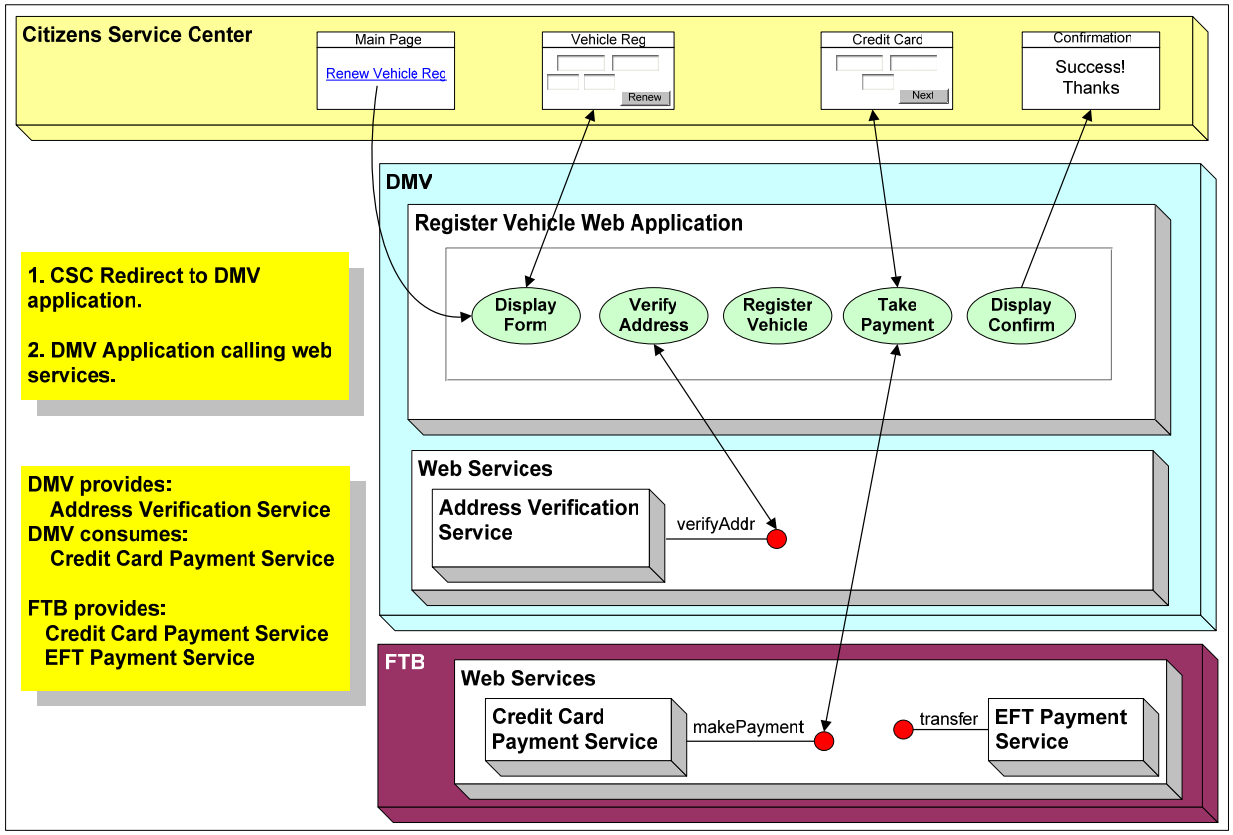
Because business processes vary greatly in their complexity, mechanisms are needed to arrange web services in different ways to achieve different business purposes. Therefore, they can be as fine-grained or course-grained as needed.

Web services can be reassembled into new services (composite),

A good web services development tool should make this relatively easy. It should be able to handle atomic, composite, federated, and orchestrated processes.

Atomic Web Services Pattern

A user is directed to the Register Vehicle Web Application at DMV. Function modules within this application directly invoke (consume) Address Verification and Credit Card Payment web services. FTB is a service *provider* with the Credit Card Payment and EFT Payment services. DMV is both a *provider* and *consumer* of the Address Verification service.



Atomic Web Services Pattern

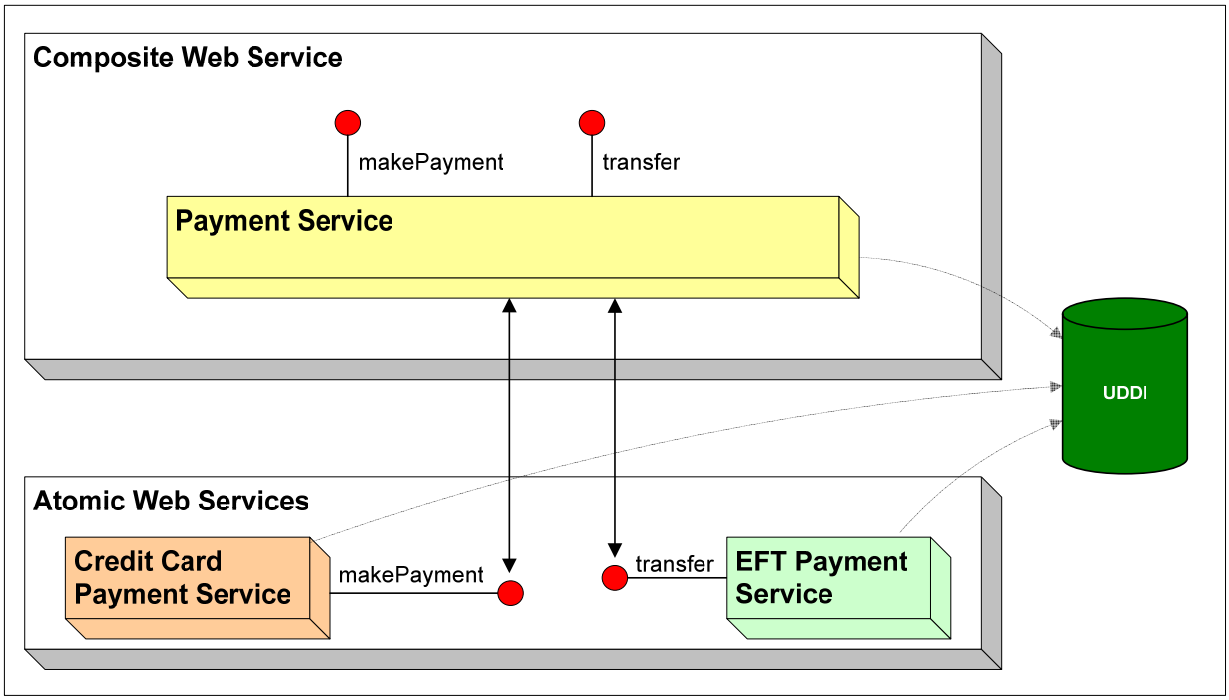
Other web applications might consume Address Verification, Credit Card Payment, and EFT Payment services. In fact, they would be registered with a UDDI repository and available to any application (with appropriate policy and security credentials).

The fact that the vehicle registration application is using web services from multiple providers is completely hidden from the user as these enterprise components collaborate behind the scenes to fulfill this business service request. The customer experience is a seamless end-to-end business interaction regardless of the process interaction efforts required behind the scene.

Composite Web Services Pattern

Web services can be aggregated to form higher level, or more course-grained services. This is one of the key ways to achieve service reuse.

In the previous example, there were two *atomic* web services - Credit Card Payment and EFT Payment Services. These could be combined into a new *composite* Payment web service.

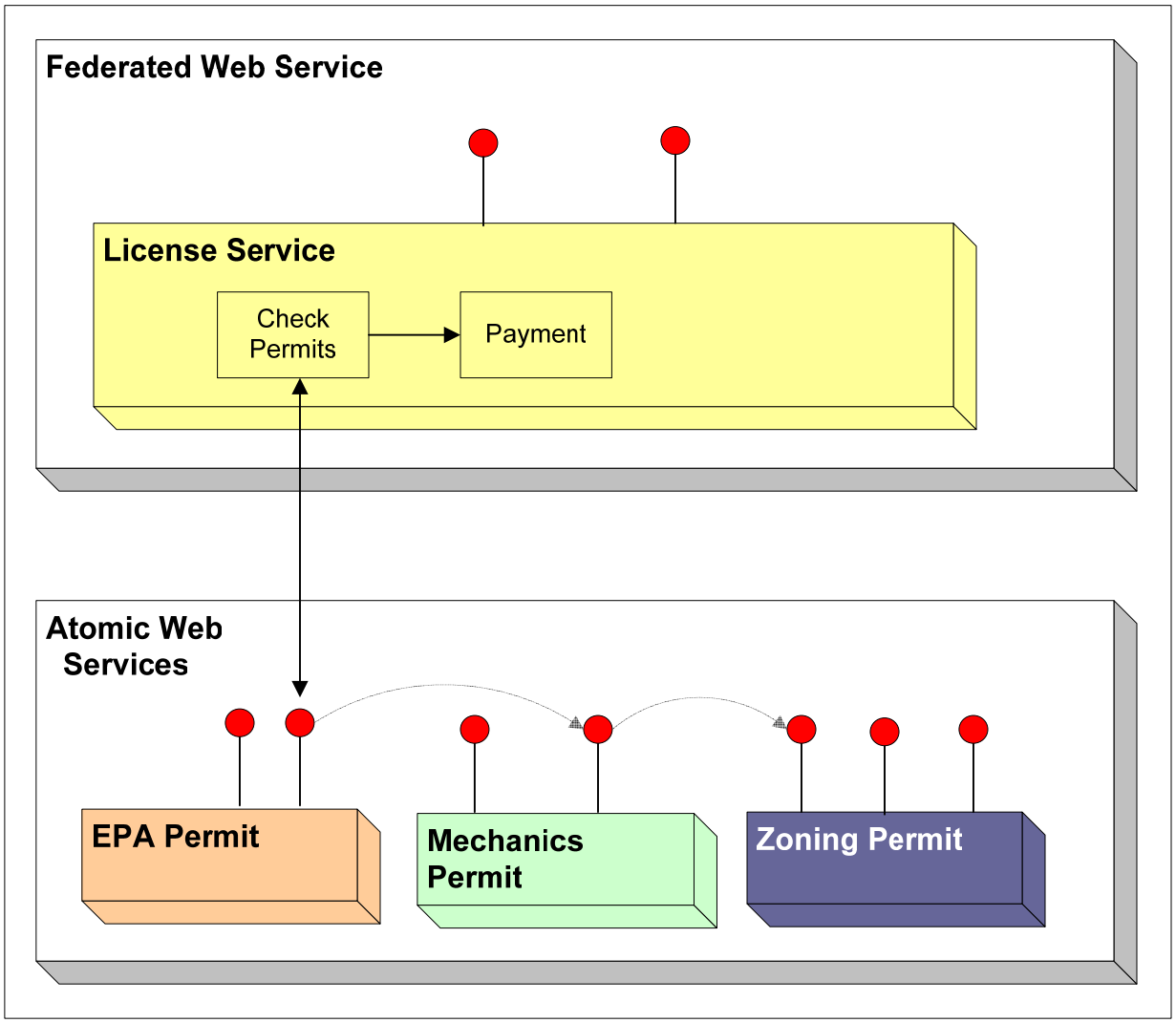


However, some business processes may only deal with credit card payments. Other, processes deal with both credit card as well as electronic fund transfer payments. So, services need to be defined in such a manner that will allow both business needs to be met. This illustrates that atomic web services can be reassembled to meet different business requirements without duplicating service code. This is one of the key principles of web services.

Federated Web Services Pattern

There are cases where a web service needs to delegate to other web services to achieve a particular part of a business process. This is similar to the composite web services pattern. However, the difference is, in a composite pattern, the atomic services are reassembled into a new service. In the federated pattern, we have a service-to-service interaction. That is, multiple interfaces from different services are combined to perform as if they were one.

In the following example, as part of processing a new license application that requires multiple purpose-specific permits, a federated License Service might do this in one step before continuing with the process. Perhaps there is a rule that states all permits must be obtained before moving to the payment process. This can be enforced by the federated service.

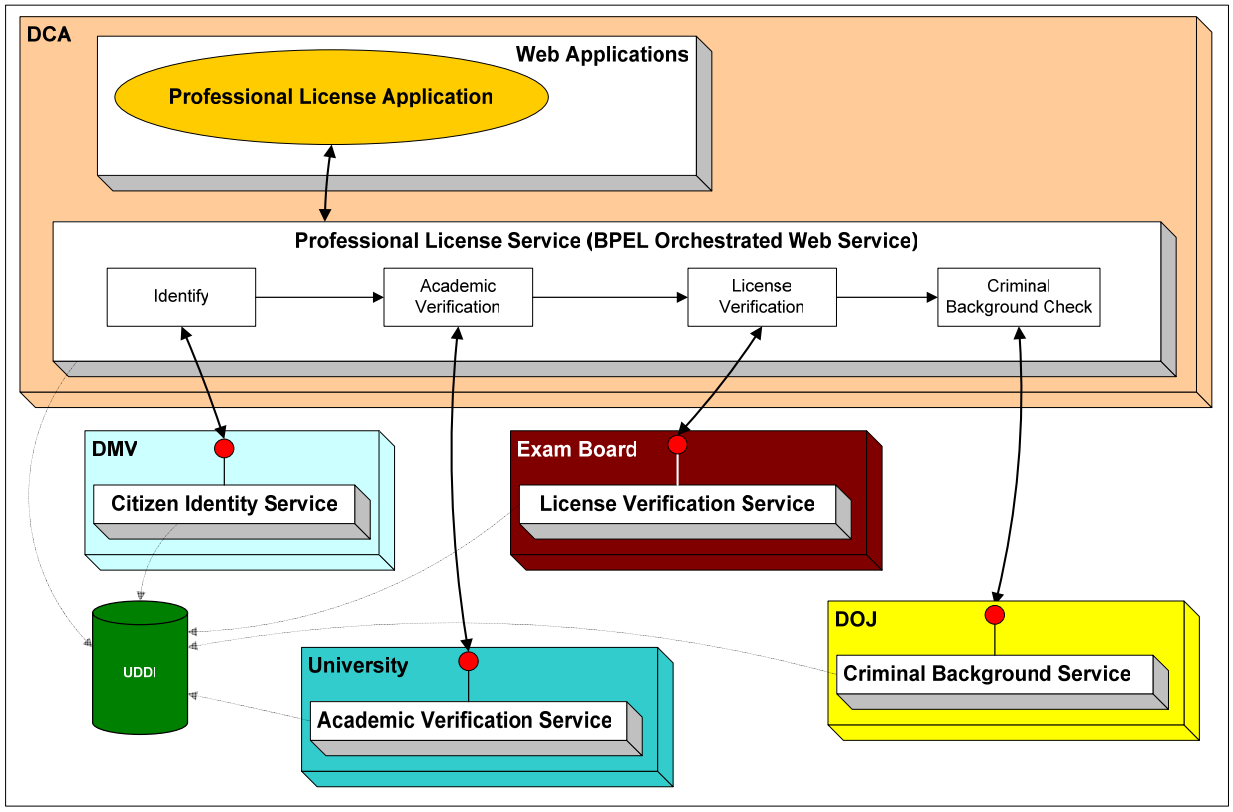


Orchestrated Web Services Pattern

In the following example, Professional License Service is an orchestrated web service that is compliant with the BPEL standard. This means one can orchestrate the flow of consumed web services. In this example, the Identity node consumes Citizen Identity Service at DMV. The Academic Verification node consumes Academic Verification Service at a university. The License Verification node consumes the License Verification Service at an exam board. Finally, the Criminal Background Check node consumes Criminal Background Service at DOJ.

One might use an orchestrated web service when there are many services to be consumed. The order in which they are consumed can be easily changed. A number of vendors are providing tools that allow a business person to determine the orchestration. Traditionally, this was an IT function particularly with workflow tools prior to web services and BPEL.

Note, atomic, composite, and federated web services can all be members of a BPEL orchestration. An orchestration can incorporate other logic such as conditional testing, navigation rules, error logic, and escalation rules if a particular service doesn't complete within the predetermined time.



Orchestrated Web Services Pattern

Enterprise Search Service Pattern

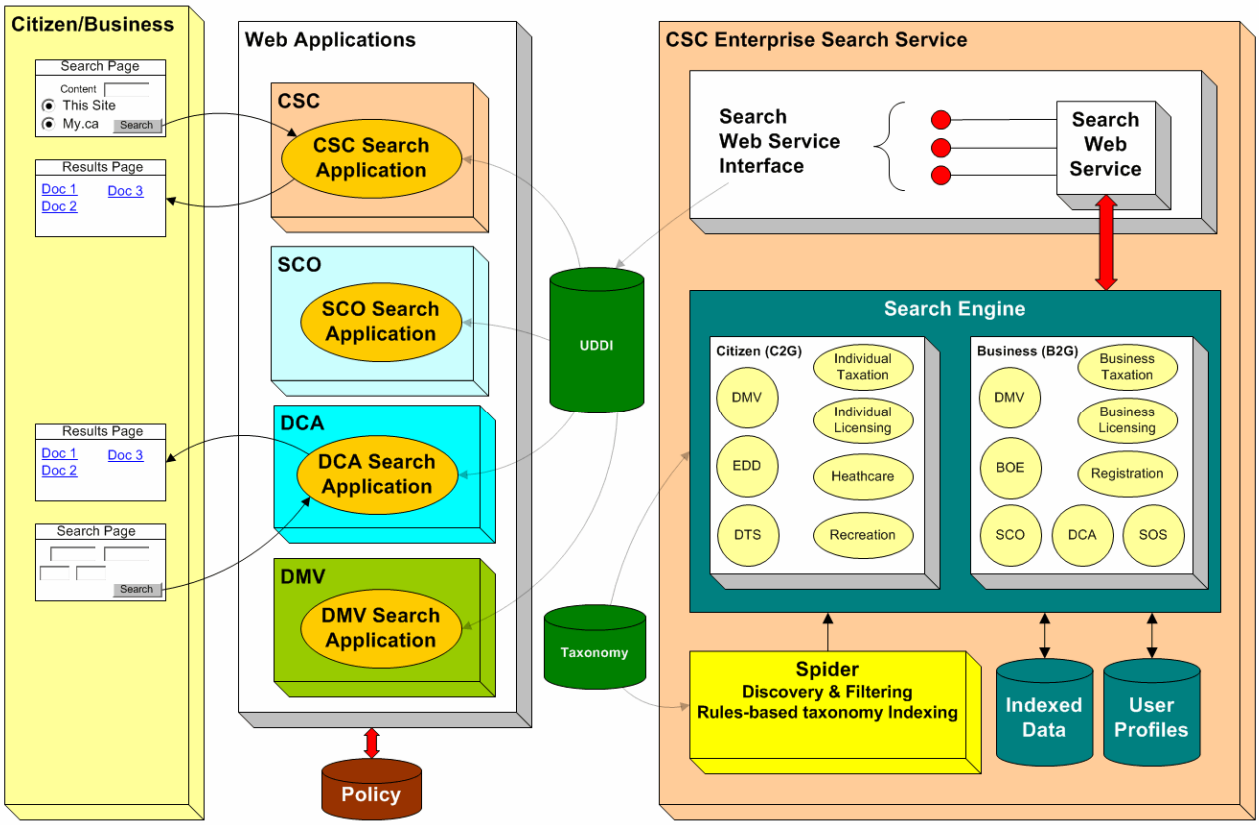
The Enterprise Search Service is a more complex example of federated services. The current State search engine should be redesigned to handle a variety of different search requests while returning more meaningful results. That is, results that meet customer expectations more closely.

In the next example, a web services interface could be utilized in conjunction with a search engine. This interface would expose the types of searches that any department application could implement. So, consistency across departments would be achieved as it would appear from the user perspective as if they were interacting with a single search application.

In fact, the search engine could be configured to index information in a more useful manner such as by user type (citizen vs business). Additionally, filtering could be applied to the spider process to not index certain content (for example, those without titles, or those defined in duplicate locations).

If the state came up with a common language (taxonomy) for defining content, then the results of the spider could be indexed according to the taxonomy.

This is an example of a federated services approach since the indexes are maintained at DTS, but the actual content is located and maintained in each department.



Enterprise Search Service Pattern

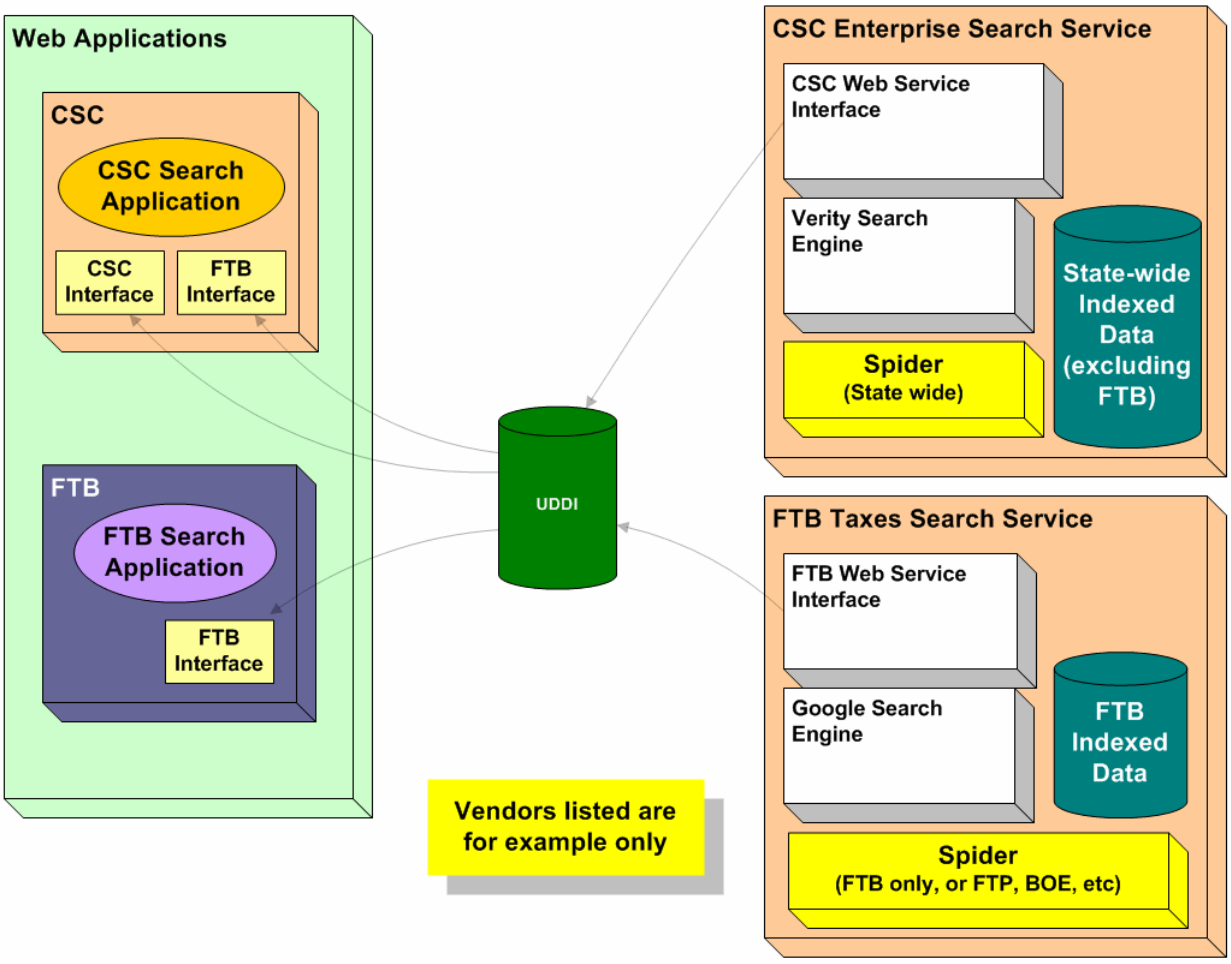
Search applications could use profile and policy information to influence search results. Additionally, search applications can use different parts of the search interface, as well as aggregate or filter the results. So, the user experience would be very different. A search application could ask the search engine (via the search interface and broker) to only look in certain collections or to use certain parts of a taxonomy.

Federated Search Engines Pattern

The above example illustrates a centralized search engine approach. Economies of scale, lower infrastructure costs, consolidated databases, lower spider traffic, and lower staff costs are some of the reasons to use this approach. It is particularly effective as a general state-wide search service that, if architected properly could intelligently defer to topic specific department search engines.

Departments maintain knowledge and expertise that can be captured in their own search engines. For example, it might make business sense for Franchise Tax Board to maintain a Tax Search Engine where one could go to get answers to any tax question. DHS might maintain a Health Search Engine, DOJ a Criminal & Justice Search Engine, etc.

Depending on the scope of the Tax Search engine, FTP might spider just their own site or they could spider FTP, BOE, and other sites that also deal with taxes. It is likely that this search engine would return a more comprehensive result set than the state-wide search engine. FTB would define a web service interface for their search engine and register this interface with the UDDI repository. Then, any search application would have the choice of implementing the CSC or FTB search interface (or both interfaces).



Federated Search Engines Pattern

In the above diagram, CSC could implement both interfaces, and based on the users query CSC could federate to FTB if the user is looking for tax information. If both search engines are from the same vendor, this capability may be built in. That is, logic to determine which search engine is in the best position to service the query is determined by search engine collaboration.

RSS (Real Simple Syndication) Pattern

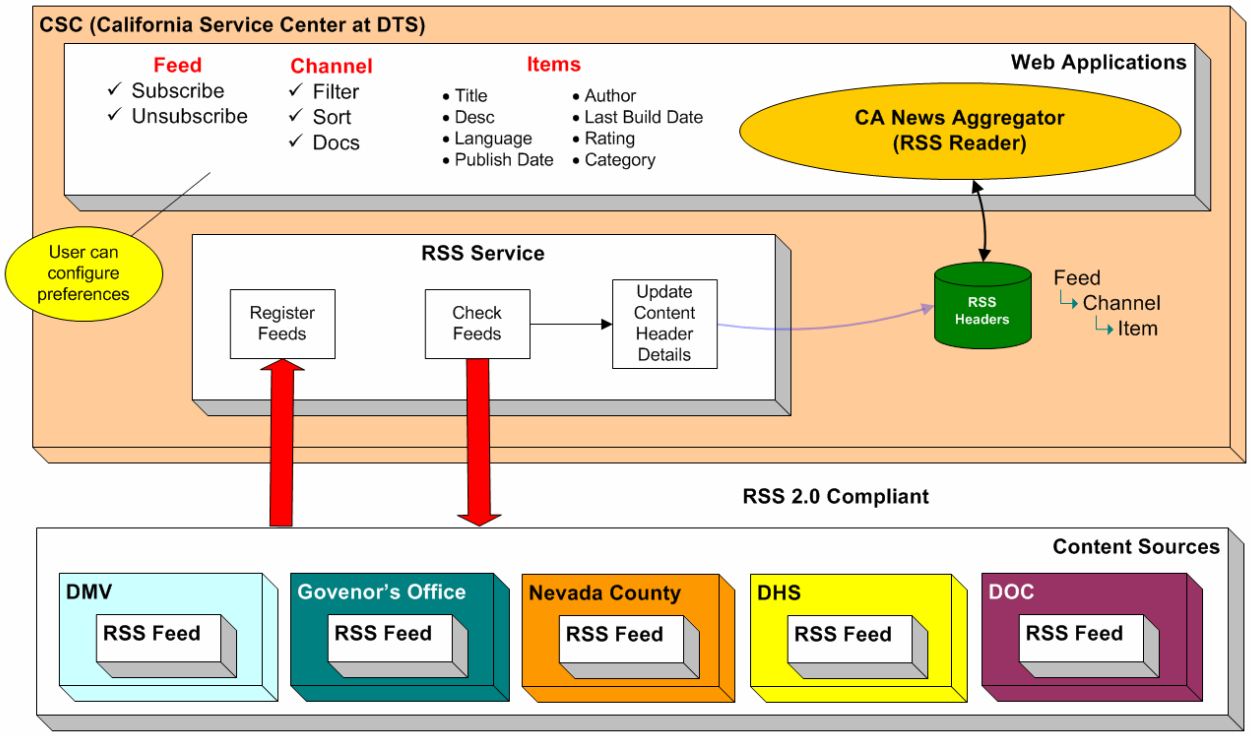
RSS is a very popular method of publishing regularly updated information. It was originally created by Netscape and used primarily by news and media companies as a way of syndicating news. When Netscape lost interest, Userland Software picked it up and continued to enhance it. A parallel effort using a different format was initiated by the W3C standards body. Fortunately, the formats have merged with RSS 2.0. It is now widely used by many companies and public entities as an easy way to keep subscribed customers informed of changes.

A couple of public sector examples are the states of Virginia and Utah. Virginia provides 34 different feeds on topics like Featured Sites, Emergency Notifications, Press Releases, Citizen Services, Online Services, Family Services, Educational Services, Government Resources, License and Permits, Forms,

and Business Services. Changes to any of the above topics are placed in the appropriate RSS file which is registered with organizations that monitor RSS feeds.

Anyone can download a free RSS Reader (called a News Aggregator), and subscribe to any RSS feed directly or to any of the many published sites that allow an individual to subscribe to multiple feeds. Additionally, RSS information can be pushed out to cell phones and PDA devices.

The California Service Center could provide an RSS Service that monitors the feeds from all state departments. Then, anyone could subscribe to the service and pick which feeds they were interested in and what format they would like to receive the information.



From a technical perspective, open source class libraries already exist that do most of the work. Some examples: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnexxml/html/xml02172003.asp http://www.rssdotnet.com/ http://aspnet.4guysfromrolla.com/articles/102903-1.aspx

In additional to the above Microsoft-based components, there are also many freely available products that are written in PERL, PHP, JavaScript, and Java.
Here are links to states:
Virginia http://www.rssgov.com/archives/000127.html
Utah http://www.utah.gov/

Utah sponsors an RSS workshop http://www.rssgov.com/rssworkshop.html. This site has a great list of other state RSS feeds as well as a good, detailed explanation of RSS.

Any site that displays either RSS or XML publishes an RSS feed.